

## GPIO Watchdog Bypass 测试软件使用说明

警告:本测试软件仅供测试验证本公司产品功能使用,不提供任何书面以及口头约定的任何类型保障(包括,但不限于驱动,动态库,应用程序等)!客户如需开发应用,请跟本公司售后联系获取相关开发资料文档支持,自行开发应用!

注意:部分主板需要更新 BIOS 后方可验证测试,仅支持烧录原始 BIOS 验证测试

序号	版本	调试历史	时间	调试部门
1	V2.00	首次调试	2022/10/10	软件部
2	V2.01	增加 B67X 系列项目支持(需更新 BIOS), 增加 Z370SL B330 B110 B100 B101 PICO-J3455, W580SL, H81R1,Z270DM, B420 等支持	2022/12/03	软件部
3	V2.02	修正 Gemilake 平台看门狗无法重置系统的问题	2022/12/08	软件部
4	V2.02	增加使能主板 Buzzer 功能	2023/01/04	软件部
5	V3.00	增加 Fintek IO Watchdog 支持	2023/06/02	软件部
6	V4.00	增加 IO 空间访问控制	2023/12/13	软件部
7	V4.00	修正 NCT6412 看门狗异常触发的问题	2024/03/11	软件部

图 1: 测试版本更改历史

序号	版本	支持 BIOS ID 项目	备注
1	V2.02	D41SL2NX , M41E1NOX, Nomia2LX, C365H.WB-803.XX, H510M16X, R730.P226-IO.XX, R730.M2XX-IO.XX, E20R2LOX, C20.P1061-R6.XX, C110.N53-C2.0X, Q570SL2NXX, PPC10VOX, C58.C57-H2.XX(C58系列通用), C57.C57-H2.XX(C57系列通用), B6712LXX, B6706LXX, B6722LXX, Z370SL-EX2XX, B3306LXX, B110XLXX, B100XLXX, B1012LXX, R67.M2XX-IO.XX, B6412LXX, PJ34552LXX, W580SL2NXX, H81R16NXX, Z270DM-EX6XX, B4202LXX, B4116LXX, E75R2LXX, B3466LXX, B75R16XX, E6416LXX, E6426LXX, N29RXLXX, B190XLXX, B413XLXX, SN10H.NX1-XX, B4214LXX, E41R6LXX, T41R2LXX, C8X.C57-H2.XX(C81 82系列通用), C100.C57-H2.XX(C100系列通用), E81R6LXX, E1006LXX, B81R6LXX,Y0I6412XX, P81.P1081-C6.XX, LS4304LXX, C641.C58-C6.XX, R270.R270-G6.XX, G370.G370-C6.XX, P122.P68-HV-G16.XX, B6832LXX, B120V16LXX, P122.P68-L6-G16.XX, C43.C58-C6.XX	简写 BIOS ID 表示 N29 需要 BIOS 版本为 03 的版本, N10 包含 N08 项目, 增加 R730.M2XX-IO 网卡 bypass 功能

图 2: 支持测试项目列表

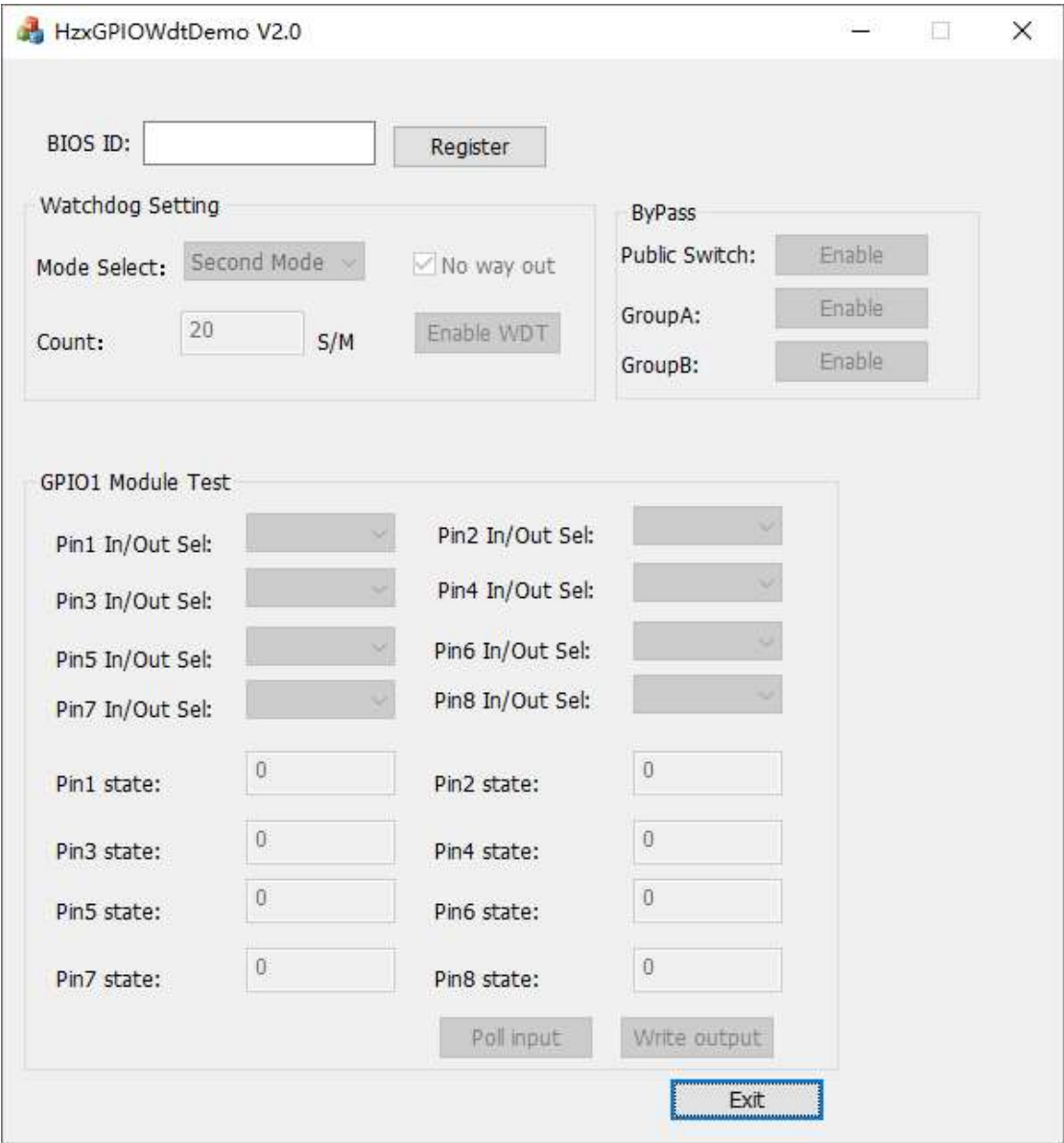
### 一. 测试软件包文件

- |                           |               |
|---------------------------|---------------|
| 1. HzxWdtGpioDemo_x64.exe | : 64 位系统可执行文件 |
| 2. libloCtrl.h            | : sdk 库头文件    |

- |   |                                    |
|---|------------------------------------|
| 3. libloCtrlx64.dll libloCtrlx64.lib    | : 64 位 release 版本动态库               |
| 4. libloCtrlx64d.dll libloCtrlx64d.lib  | : 64 位 debug 版本动态库                 |
| 5. Wbhwdoct64.dll                       | : 64 位系统动态库                        |
| 6. WBHWD OCT64.sys                      | : 64 位系统硬件驱动                       |
| 7. VC2019_redist.x64.exe                | : visual studio 2019 64 位系统运行时库安装包 |
| 8. HzxGPIOWdtDemo-230808.zip            | : 测试软件源代码                          |
| 9. GPIOWatchdogBypass 测试软件使用说明.pdf      | : 本文档                              |
| 10. libloCtrlx86.dll libloCtrlx86.lib   | : 32 位 release 版本动态库               |
| 11. libloCtrlx86d.dll libloCtrlx86d.lib | : 32 位 debug 版本动态库                 |
| 12. Wbhwdoct.dll                        | : 32 位系统动态库                        |
| 13. WBHWD OCT.sys                       | : 32 位系统硬件驱动                       |
| 14. GPIONTest-221011.zip                | : 64 位系统 C#简单例程                    |

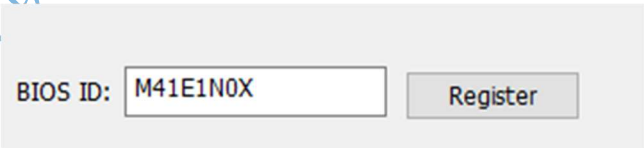
## 二. 测试前准备

1. 安装 Microsoft Visual Studio 运行库: VC2019\_redist.x64.exe
2. 重启系统后进入测试文件夹, 右键点击 HzxWdtGpioDemo\_x64.exe, 选择“以管理员身份运行”, 进入以下类似界面:



注意：本示例测试程序以 M41E 项目为测试主板

在 BIOS ID 栏填写项目识别的 BIOS ID(参考支持测试项目列表)，测试 M41E 项目，填写“M41E1N0X”，点击“Register”如果受支持，或者没有异常，将启动响应的测试模块，如：



(1) 看门狗测试模块(Watchdog setting)

Mode Select: 秒/分钟模式选择

Count: : 看门狗定时器时间

No way out: 选择，不可关闭看门狗功能，不选择，可以关闭看门狗功能

在选择 No way out 并点击 Enable WDT 后退出测试软件，看门狗将在定时器剩余时间到达后硬

重启主板

## (2) GPIO 测试模块(GPIO1 Module Test)

Pin(x) In/Out Sel : 选择管脚(x)为输入或者输出 x 为 1,2,3,4,5,6,7,8

Pin(x) state : 管脚(x)的当前输入状态(仅为 Pin(x) In/Out Sel 为 Input 的管脚有效, 为 output 的管脚为可写状态)

Poll input: 循环查询管脚为输入脚的当前状态(点击选择) ---- 0 为低电平, 1 为高电平

Write output: 写入指定状态到管脚为输出脚的管脚(点击选择) ---- 0 为低电平, 1 为高电平

Exit: 选择 OK 将不退出软件而缩小到托盘区。选择 cancel 将退出测试软件

## (3) Bypass 测试(需主板支持), 当前 Public Switch 无效

GroupA: 打开或者关闭

GroupB: 打开或者关闭

注意: 具体的控制组别关系, 请跟硬件工程师确认

## 三. 测试源代码重编译:

注意: 设备在同一时刻只允许一个线程访问, 请注意同步线程, 本代码仅供测试验证使用

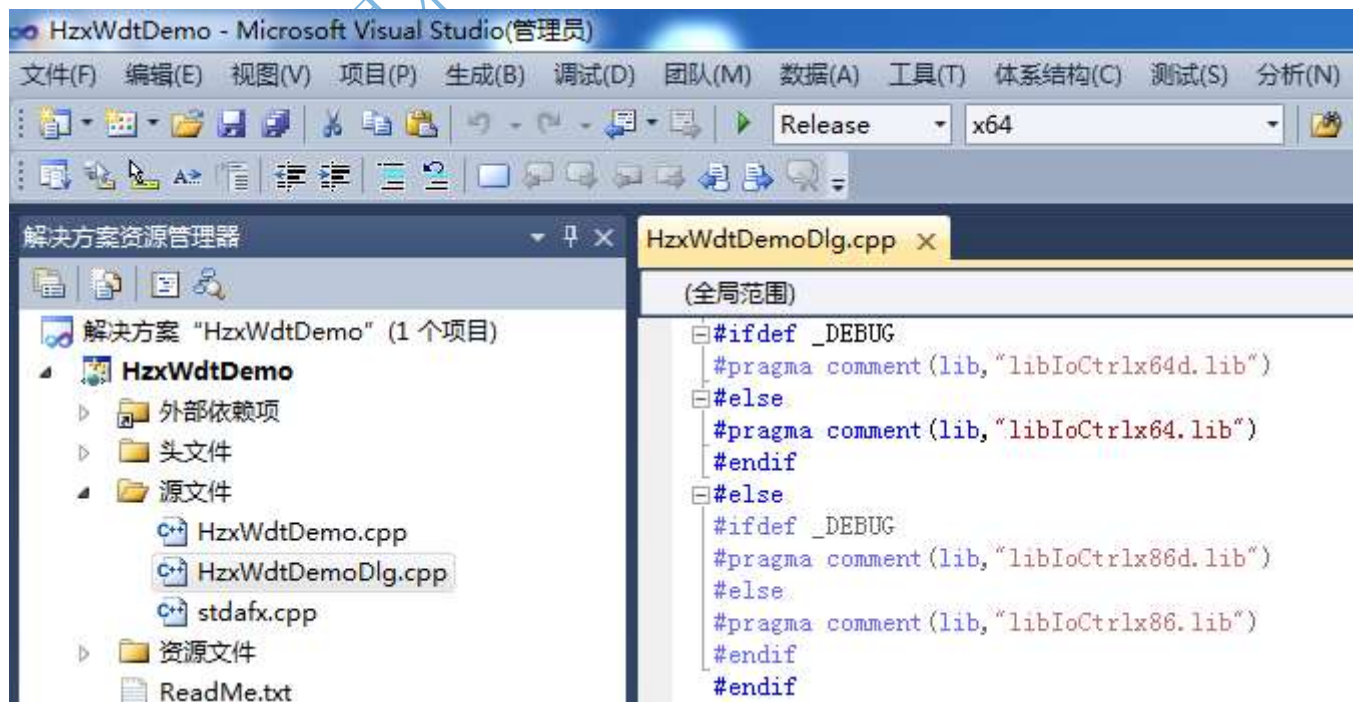
开发环境为 windows 10 64 位

Microsoft Visual Studio 2019 MFC x64

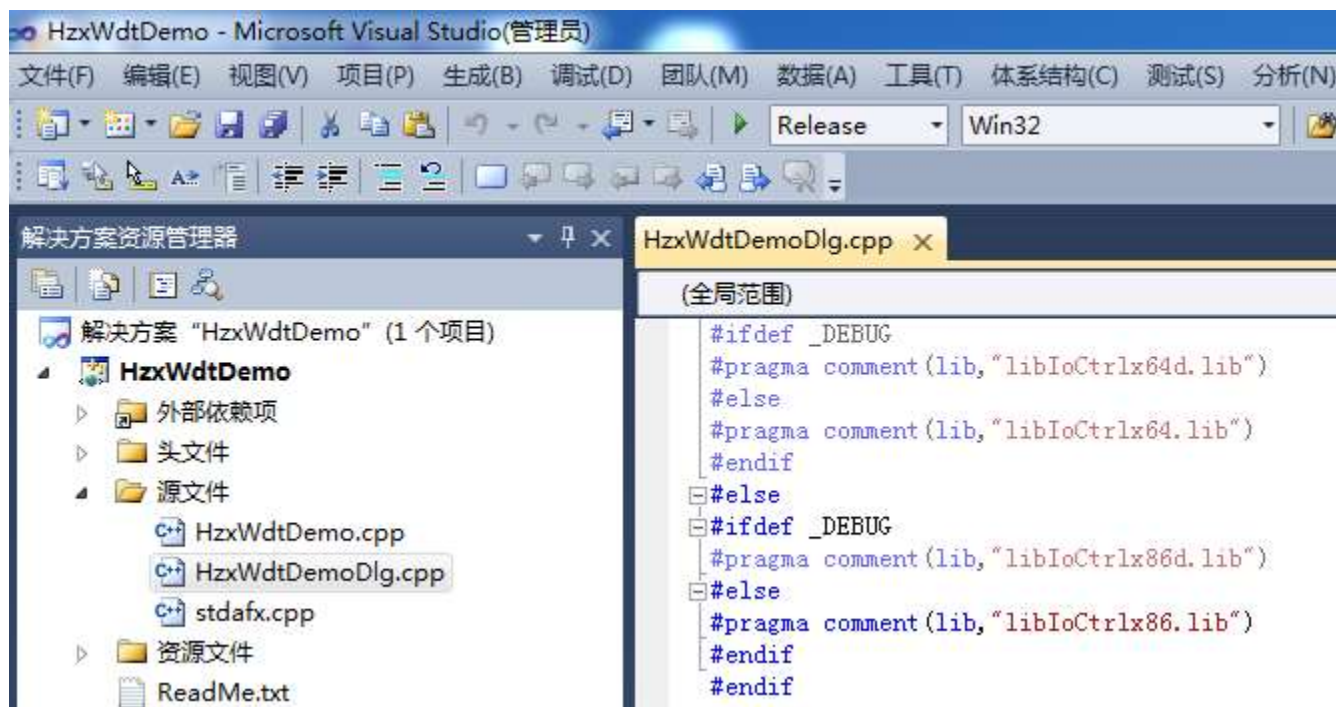
## 1. Windows 32位系统编译配置

将编译环境从x64更改为Win32即可。如下所示:

从



## 更改为



#### 四. 测试程序使用动态库的流程

头文件为libIoCtrl.h, x64链接库文件为libIoCtrlx64d.lib (debug), libIoCtrlx64.lib (Release)

必需的运行时文件清单(x64): libIoCtrlx64.dll (Release), libIoCtrlx64d.dll (debug)  
Wbhwdoct64.dll WBHWDOCT64.sys

Win32链接库文件为libIoCtrlx86d.lib (debug), libIoCtrlx86.lib (Release)

必需的运行时文件清单(Win32): libIoCtrlx86.dll (Release), libIoCtrlx86d.dll (debug)  
Wbhwdoct.dll WBHWDOC.sys

当前动态库版本为V2.0。

#### 1. 调用的依次顺序(函数输入参数可参考测试程序源代码HxzGPIOWdtDemo及libIoCtrl.h头文件)

(1) 看门狗测试时:

libIoCtrlInit-> OpenWatchdog-> updateWatchdog(可选)-> enableWatchdog(可选)-> KeepWatchdogAlive->CloseWatchdog->libIoCtrlDeinit

(2) GPIO测试时

libIoCtrlInit-> getPinLevel 或者 setPinLevel-> libIoCtrlDeinit

(3) Bypass测试:

libIoCtrlInit-> IsLanBypassSupported-> LanBypassEnable-> libIoCtrlDeinit

#### 2. 头文件libIoCtrl.h说明

```

typedef enum{
    GPIO_PIN1, ->GPIO管脚1
    GPIO_PIN2, ->GPIO管脚2
    GPIO_PIN3, ->GPIO管脚3
    GPIO_PIN4, ->GPIO管脚4
    GPIO_PIN5, ->GPIO管脚5
    GPIO_PIN6, ->GPIO管脚6
    GPIO_PIN7, ->GPIO管脚7
    GPIO_PIN8 ->GPIO管脚8
    .....
} GPIO_PIN_INDEX;

typedef enum{
    GPIO_LOW_LEVEL, ->GPIO管脚输出低电平
    GPIO_HIGH_LEVEL ->GPIO管脚输出高电平
} GPIO_LEVEL;

typedef enum{
    WDT_SECOND_MODE, -> 看门狗为秒模式
    WDT_MINUTE_MODE ->看门狗为分钟模式
} WDT_TIMER_MODE;

struct __WDT_INFO{
    int nowayout; ->为1时，看门狗模块不可关闭，为0时，看门狗可以关闭
    unsigned char mode; -> 看门狗模式，为枚举量---- WDT_TIMER_MODE
    unsigned char time;->看门狗定时器时间
    unsigned char enable;->1 配置同时使能看门狗(默认) 0 :仅配置看门狗，不使能(不
推荐)
};

/*
**函数: libIoCtrlInit
**函数说明:初始化驱动及导入SDK动态库
**参数: hlib 模块地址指针, bios_id 主板BIOS ID号, 在主板BIOS setup中可以查看
得到、
**返回值: 0 成功, 非0失败 , 成功时并导出HMODULE的实例
*/
extern "C" __declspec(dllexport) int libIoCtrlInit(HMODULE *hlib, char*
bios_id);

/*

```



```

**函数: libIoCtrlGetVersion
**函数说明: 获取SDK动态库版本
**参数: hlib 模块地址指针, version Sdk版本号
**返回值: 0 成功, 非0失败, 成功时导出demo sdk 的版本号(字符串)
*/
extern "C" __declspec(dllexport) int libIoCtrlGetVersion(HMODULE *hlib, char*
version);

//for gpio
/*
**函数: setPinInOut
**函数说明: 配置管脚的输入输出
**参数: hlib 模块地址指针, index 管脚序号 (参看GPIO_PIN_INDEX枚举定义), state
管脚输入输出设置(参看GPIO_DERECTION_INFO的枚举定义)
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int setPinInOut(HMODULE *hlib, GPIO_PIN_INDEX
index, GPIO_DERECTION_INFO state);

/*
**函数: getPinLevel
**函数说明: 获取管脚的输入输出状态
**参数: hlib 模块地址指针, index 管脚序号 (参看GPIO_PIN_INDEX枚举定义),
curstate 导出的管脚输入输出状态, 0为低电平, 1为高电平
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int getPinLevel(HMODULE *hlib, GPIO_PIN_INDEX
index, unsigned char* curstate);

/*
**函数: setPinLevel
**函数说明: 设置管脚的输入输出状态
**参数: hlib 模块地址指针, index 管脚序号 (参看GPIO_PIN_INDEX枚举定义),
curstate 设置管脚输入输出状态(参看GPIO_LEVEL枚举定义), GPIO_LOW_LEVEL为低电平,
GPIO_HIGH_LEVEL为高电平(仅为寄存器状态, 实际电路的状态, 取决于外部电路)
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int setPinLevel(HMODULE *hlib, GPIO_PIN_INDEX
index, GPIO_LEVEL curstate);

//for watchdog
/*
**函数: OpenWatchdog

```

```

**函数说明:打开看门狗测试模块并初始化
**参数: hlib 模块地址指针, wdtinfo 看门狗状态定义, 请参看struct __WDT_INFO的
定义
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int OpenWatchdog(HMODULE * hlib, struct
__WDT_INFO* wdtinfo);

/*
**函数: updateWatchdog
**函数说明:打开看门狗测试模块并初始化
**参数: hlib 模块地址指针, wdtinfo 看门狗状态定义, 请参看struct __WDT_INFO的
定义
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int updateWatchdog(HMODULE * hlib, struct
__WDT_INFO* wdtinfo)

/*
**函数: enableWatchdog
**函数说明:启动看门狗, 这个参数取决于struct __WDT_INFO 的nowayout的配置, 如
果nowayout为1时, 此函数无效, 任何时候都会返回成功, 但是并不会做开启或者关闭动作
**参数: hlib 模块地址指针, enValue 1 打开 0 关闭
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int enableWatchdog(HMODULE * hlib, unsigned
char enValue);

/*
**函数: KeepWatchdogAlive
**函数说明:喂狗函数, 以OpenWatchdog或者updateWatchdog
**          的配置参数为准
**参数: hlib 模块地址指针,
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int KeepWatchdogAlive(HMODULE * hlib);

//Lan Bypass support

/*
**函数: IsLanBypassSupported
**函数说明:查询项目是否支持网络bypass功能
**

```



```

**参数: hlib 模块地址指针
*      isSupport 导出项目支持lanbypass状态,0为不支持, 1为此项目支持Lan
Bypass
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int IsLanBypassSupported(const HMODULE * hlib,
unsigned char* isSupport);

/*
**函数: GetLanBypassState
**函数说明: 查询Bypass支持组的状态
**
**参数: hlib 模块地址指针
*      state 导出lanbypass组状态, bit0为Group A的状态, bit1为Group B的状态 位
值为0表示目前为关闭状态, 1为开启状态
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int GetLanBypassState(const HMODULE * hlib,
unsigned char* state);

/*
**函数: LanBypassEnable
**函数说明: 开启或者关闭Lan bypass
**
**参数: hlib 模块地址指针
*      index lanbypass组序号 (参考BYPASS_INDEX)
*      Enable 0: 关闭 1: 打开
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int LanBypassEnable(const HMODULE * hlib,
BYPASS_INDEX index, unsigned char Enable);

//deinit system driver,exit
/*
**函数: libIoCtrlDeinit
**函数说明: 卸载驱动及导入SDK动态库
**参数: hlib 模块地址指针
**返回值: 0 成功, 非0失败
*/
extern "C" __declspec(dllexport) int libIoCtrlDeinit(HMODULE *hlib);

```